December 1997 GMU/C3I-195-IR

CENTER OF EXCELLENCE IN COMMAND, CONTROL, COMMUNICATIONS AND INTELLIGENCE

GEORGE MASON UNIVERSITY
Fairfax, Virginia 22030
ANNUAL TECHNICAL REPORT
for the period
1 July 1997 - 31 October 1997
for

ADAPTIVE DECISION MAKING AND COORDINATION IN VARIABLE STRUCTURE ORGANIZATIONS

Grant Number N00014-93-1-0912

C3 Architectures Laboratory
Center of Excellence in Command, Control, Communications, and Intelligence (C3I)

George Mason University
Fairfax, Virginia 22030

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1997	**Technical Report,** 1/1/97 - 10/31/97

4. TITLE AND SUBTITLE

ADAPTIVE DECISION MAKING AND COORDINATION IN VARIABLE STRUCTURE ORGANIZATIONS

5. FUNDING NUMBERS

N00014-93-1-0912

6. AUTHOR(S)

Alexander H. Levis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Center of Excellence in Command, Control, Communications and Intelligence
George Mason University
Fairfax, Virginia 22030

8. PERFORMING ORGANIZATION REPORT NUMBER

GMU/C3I-195-IR

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Progress in research on coordinations in distributed organizations with variable structure is reported. A genetic algorithm approach for designing variable structure organizations is presented The effort to support model driven experimentation for Adaptive Architectures in Command and Control is described.

14. SUBJECT TERMS

Decision Making; Organization; Colored Petri Nets; Genetic Algorithms

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	**Unclassified**	**Unclassified**	**UL**

CENTER OF EXCELLENCE IN
COMMAND, CONTROL, COMMUNICATIONS AND INTELLIGENCE

GEORGE MASON UNIVERSITY
Fairfax, Virginia 22030

ANNUAL TECHNICAL REPORT

for the period

1 January 1997 - 31 October 1997

for

ADAPTIVE DECISION MAKING AND COORDINATION IN VARIABLE STRUCTURE ORGANIZATIONS

Grant Number N00014-93-1-0912

Submitted to
Dr. W. S. Vaughan, Jr. (3 copies)
Office of Naval Research
Code 342
800 North Quincy Street
Arlington, Virginia 22217-5000

Copies to:
Director, Naval Research Laboratory
Administrative Grants Office, ONR
Defense Technical Information Center

Submitted by:

Alexander H. Levis
Principal Investigator

December 5, 1997

TABLE OF CONTENTS

LIST OF FIGURES

1. PROGRAM OBJECTIVES

The objective of this research, as described in the 1993 and 1996 proposals and the previous progress reports, is the investigation of several issues related to adaptive architectures for command and control and the coordination necessary not only for the functioning of an organization, but also for managing the adaptation. In particular, an organization is coordinated through direct and indirect means. The direct means includes the set of decision rules that the organization members use and the commands that they issue to each other. Indirect means include the dissemination of information within the organization; for example, organization members may share information or they may inform each other as to the actions they plan to take or decisions they have made. Coordination becomes a complex issue in adaptive organizations; organizations that adapt their architecture to changes in the tasks, changes in the environment, or changes in the available resources. Not only do the decision rules and the information architecture have to work for each fixed structure, but the designer has to deal with the problem, a metaproblem, of coordinating the variability. This becomes a particularly difficult problem in organizations that exhibit substantial complexity and redundancy in their information structure. The redundancy is necessary both for robustness and for flexibility and reconfigurability. In order to address these problems a set of tasks were defined in the 1993 and 1996 proposals; some additional tasks were defined in the modifications that took place in 1995; they are described in the next section. In addition, some basic work in algorithms and Colored Petri Nets needs to be done to develop tools and techniques for supporting the analysis and design.

2. STATEMENT OF WORK

The statement of work, as outlined in both proposals and several modifications, is given below. Given that the period of performance covered by the 1993 proposal has ended, several tasks were closed and were replaced by the tasks defined in the 1996 proposal. This will be indicated in the Research Plan (Section 3) and the Status Report (Section 4).

Task 1 (A): Consistency and Completeness in Distributed Decision Making

Develop a methodology for analyzing and correcting the set of decision rules used by an organization with distributed decision making. The methodology is to be based on the modeling of the set of decision rules in the form of a Colored Petri Net and on the analysis of the net using S-invariant and Occurrence graphs. The ability to verify and correct the set of decision rules has direct impact on the extent of coordination needed in an actual organization and the resulting communication load.

Task 2 (A): Variable Structures: Heuristic Rules in the Lattice Algorithm Constraints

Develop a methodology for considering additional constraints in the Lattice Algorithm. Such constraints include the degrees of redundancy and complexity at the different processing nodes (to be derived from the DFS algorithm of Andreadakis), the projected response time of the organization, ands some user-specified constraints on connections between decision making units. Develop a procedure for checking the validity of such constraints and incorporate them in the Lattice Algorithm. Generalize the approach to multilevel organizational structures and to variable structures, where variable structures are obtained by folding together different fixed structures. The real focus of the task is to introduce these additional constraints as a way of containing the dimensionality problem inherent in flexibility and reducing the coordination requirements.

Design a symbolic interface for the Lattice algorithm. The interface would have the capability of interpreting natural language inputs entered by the user and will include some symbolic processing. The system will generate the interconnections matrices used as input to the

Lattice algorithm. The designer would then use the various tests described in the proposal (such as DFS algorithm) to check the validity of the interconnection constraints and to make required modifications.

Task 3 (A): Dynamic Task Allocation in Adaptive C2 Architectures

The objective of this task is the application of CAESAR II (Computer-Aided Evaluation of System Architectures) to several problems associated with the organizational design of flexible Command and Control Architectures for Joint Operations in modern littoral warfare. Specifically, the tools and techniques developed for the design of distributed tactical decision making organizations and for designing adaptive information structures for such organizations are embodied in the software suite in CAESAR II. When a well defined organizational task is mapped onto the humans and machines constituting the organization, several problems of inconsistency, redundancy, and ambiguity arise that degrade organizational performance. The results of Task 1 - theory and algorithms - will be applied to this problem. Furthermore, the existence of CAESAR II with the enhancements of Task 2 makes possible the support of model-driven experimentation.

Task 4 (A): Graphical Representation of C2 Decision Making and Supporting Inference

The objective of this task is to use the rapidly developing field of influence diagrams and Bayesian networks to build graphical representations of decision making and supporting inference (intelligence) processing for command and control organizations. In particular, the emphasis is on distributed command and control organizations that are involved in rapidly evolving tactical situations and are likely to reorganize so as to remain effective. As this is our first effort in this area, our focus for this task will be on the representation of decision making and inference processes within command and control. The representation issues that we will address are distributed, concurrent, asynchronous, and interconnected command and control elements; the evolution of decision making tasks throughout a typical tactical mission; the exchange of information between elements associated with the evolution of time within a specific mission phase; and the impact of overarching the environmental and threat uncertainties that inhibit the effective partitioning of the command and control elements. This activity will lead to critical new representation ideas in influence diagrams and Bayesian networks, enabling later research that addresses partitioning algorithms of decision making and inference tasks for the purpose of effective allocation of such tasks to command and control elements.

Task 5 (B): Distributed Process Coordination in Adaptive C2 Teams

5.1 Develop further and refine the algorithm for the derivation of coordination rules that support alternative back-up strategies.

As part of a current research task, an algorithm has been outlined for the derivation of the coordination strategies of decision makers in adaptive command and control teams. This is accomplished by utilizing a Colored Petri Net representation of the decision making organization.

5.2 Develop an intuitive user interface for this algorithm and incorporate it in the suite of tools that constitute CAESAR II.

CAESAR II is a suite of tools that support the analysis, design, and evaluation of organizational architectures. The developed algorithm will be incorporated in CAESAR II and a graphical user interface will be developed for the user to interact with the algorithm and specify either a predefined back-up strategy or formulate new ones.

5.3 Apply the algorithm to several realistic examples using a variety of back-up strategies.

Examples drawn from the Adaptive Architectures for Command and Control (A2C2) initiative will be used to test the algorithm and analyze the results.

Task 6 (B): Behavioral and Performance Evaluation of Adaptive Architectures

6.1 Extend the System Effectiveness Analysis methodology to include adaptive architectures for Command and Control.

The System Effectiveness Analysis methodology has been developed and applied to fixed structure systems, including organizations with fixed structure. Basic research is needed to extend the methodology to adaptive architectures.

6.2 Incorporate in CAESAR II the analytical and graphical tools (using COTS to the maximum extent possible) necessary to apply the methodology to the assessment of adaptive organizations.

6.3 Develop the analytical and computational constructs necessary to assess the sensitivity of the effectiveness measure to the design parameters that determine the System Locus and the mission parameters that determine the Requirements Locus

Task 7 (B): Modeling Decision Making Activities for Adaptive C^2 Architectures

7.1 Extend graphical model language for specifying Command and Control planning tasks to address both the time sequencing of plans and concurrent battle execution and battle damage assessment (BDA).

This task extends early work on modeling decision making in a distributed, hierarchical organization by combining influence diagrams and Bayesian networks with Petri Nets. To date, we have decomposed the difficult decision making tasks solved by joint task forces to address the needed interaction amongst the decision making tasks at a given level and across levels of the organizational structure. We will address the complex decision making task models needed to define concurrent activities within each cell and show the effects on hierarchical, distributed decision making.

7.2 Define and analyze algorithms for assigning decision making functions to the C^2 entities of the organizational architecture.

This activity will develop measures of effectiveness for these assignments that are based upon decision making quality and timeliness. Communication issues will be treated as requirements for transmitting data elements amongst organizational entities, rather than as a measure of effectiveness.

7.3 Develop decision making structures that can be used to analyze (1) concurrent battle planning and planning for BDA, and (2) concurrent time-sequenced planning effectively.

These structures will build upon work already described as dynamic decision networks that incorporate the use of both influence diagrams and Bayesian networks.

Task 8 (A, B): Information Dissemination

Progress reports will be submitted in accordance with ONR requirements. The results of this research will appear in thesis reports and in technical papers to be presented at professional meetings and published in archival journals. In addition, oral presentations will be given periodically at review meetings organized by ONR

Tasks designated with the letter A are from the 1993 proposal; tasks designated B are from the 1996 one.

3. RESEARCH PLAN

The research plan describes the strategy for meeting the program objectives. Specifically the research plan is organized around a series of specific well-defined research tasks that are appropriate for theses at Master's and Ph.D. levels. Individual students are assigned to each task under the supervision of the principal investigator. Additional staff from the C3I Center are included in the project whenever there is a specific need for their expertise. Additional work on genetic algorithm design is now included under Task 5(B).

Task 1 (A) has been closed; a Ph.D. thesis by Abbas Zaidi was completed and the results published in an archival journal. In addition, a genetic algorithm was applied to the design of alternative organizational structures. While the Lattice algorithm generates all feasible structures given a set of constraints, it is often limited to small organizations due to the large number of alternative structures for an organization of arbitrary size. On the other hand, the lattice algorithm is not exhaustive, but can be used for the design of large organizations.

Task 2 (A) has been closed; a Ph.D. thesis by Didier Perdu was completed and the results have been published in conference proceedings and are been reviewed for publication in an archival journal. The thesis by Perdu also addresses the problem of dynamic task allocation by considering the morphing of an organization from one desired organizational form to another.

Task 3 (A) evolved into Task 7 (B).

Task 4 (A) was addressed only in part. A successful effort was made to improve earlier work on the conversion of an influence diagram into an executable Petri Net. Two algorithms were developed: one for taking an influence diagram developed using the COTS package Analytica™ and one that transforms an influence diagram developed using SIAM (Situational Influence Assessment Module).

The research plan now consists of Tasks 5 to 8. Progress on these tasks is reported in Section 4.

4. STATUS REPORT

The earlier work on the use of genetic algorithms for the design of decision making organizations, reported in the previous annual report, addressed fixed structure organizations. The new work, reported in Section 4.1, addresses variable structures. It brings in the work of Luo (1992) on coordination constraints and the work of Demaël (1994) on folding fixed structures into variable structures, both developed under earlier contracts from ONR.

Task 6 received limited attention during this year. The evaluation module in CAESAR II, named SEAT for Systems Effectiveness Analysis Tool, was recoded using MATLAB™ and became a self standing module that is used in a variety of projects - including Task 7. A description of SEAT was given in last year's report. Two theoretical efforts were initiated, one by Lee Wagenhals, a Ph.D. candidate, and one by InSub Shin, a Ph.D. Graduate Research Assistant, to extend the theory of System Effectiveness Analysis to apply to a wider class of problems. These efforts are at a preliminary stage and will not be reported in this report.

Task 7 started in 1995 and received much attention during this past year. This task is focused on applying CAESAR II to the Adaptive Architectures for Command and Control (A2C2) program. The first effort in this task was reported separately in a progress report issued in July 1996. The second effort was also reported separately in November 1997; it consisted of the model of the third experiment and the simulation results. Progress on this task is reported in Section 4.2.

4.1 DISTRIBUTED PROCESS COORDINATION IN ADAPTIVE COMMAND AND CONTROL TEAM (A. Zaidi and Levis)

4.1.1. Introduction

This report presents an extension of the work reported in the 1996 Annual Progress report on generating organizational architectures using genetic algorithms (GA). In that earlier work of Zaidi and Levis [1966], an organizational structure is viewed as an individual in a population. An initial population of organization structures is specified with the help of designer's requirements and constraints. The population of structures is enhanced genetically by means of genetic operations. The newly generated structures are tested for structural and design requirements and are assigned a fitness based on this evaluation. The feasible and/or stronger structures are retained in the population and are allowed to reproduce, while weaker ones are removed. The best individual(s) in the final population produced can be used as the solution to the design problem.

The GA-based approach just described, is used to solve the design problem for *fixed-structure* organizations. An organization has a fixed structure if the structure of the organization, and the functions each organization member performs, do not vary with the changes in the external and internal parameters. On the other hand, an organization has a *variable* structure, if the interactions between the components that belong to the organization and the process each component performs can vary in response to external or internal changes. A fixed-structure organization can, therefore, be viewed as a particular mode-of-operation of an otherwise variable-structure organization, capable of handling some predefined situation(s) determined by the internal and external parameters. Each feasible input situation to the organization is associated with a unique constituent fixed-structure in the variable-structure organization; the mode of operation (interactions, functions, and structure) of the organization is determined uniquely by the input it processes. The interactional structure of a variable-structure organization, therefore, can be viewed as a folded structure of several underlying fixed-structure organizations. (Luo and Levis, 1992) This definition of variable-structure organization translates into a design requirement: all the components of a variable-structure organization should be capable of distinguishing among different situations for which they are supposed to behave differently (i.e., execute different functions, etc.) as part of their coordination mechanisms. This requirement was first identified by Demaël and Levis (1994) in work carried out in 1989 on a formal framework for the generation of variable-structure organizations. Later, Luo and Levis (1992) presented an algorithm to check this requirement, termed Coordination Constraint, in a variable-structure organization. A variable structure which satisfies the coordination constraint is the feasible solution to the design problem.

This work combines the earlier approach of generating fixed-structure organizations by GAs with the Luo and Demaël approach of folding these fixed structures into a single variable structure and checking it for the coordination constraint. A variable organization is represented as an *organism* with several layers of *chromosomes* representing fixed-structure organizations. Populations of these organisms are produced using genetic operators and each individual in these generations is checked for feasibility using a fitness function. Information obtained during this evaluation is used in genetic operators to produce more fit individuals in the next generation. Figure 4.1 presents a top-level description of the genetic algorithm (the description in the figure describes most algorithms.) (Davis, 1991) The rest of this section describes the details of the implementation of this approach for the design of variable-structure organizations.

The section is organized as follows. Section 4.1.2 presents a mathematical model for the representation of input situations, fixed-structure, and variable-structure organizations. It also, briefly, describes the algorithm for checking the coordination constraint. Section 4.1.3 describes the proposed GA-based approach for the solution of design problem. The section also outlines

several issues that need to be investigated for a proper implementation of the approach. Finally, Section 4.1.4 concludes the discussion.

1. Initiate a Population of chromosomes.
2. Evaluate each chromosome in the population.
3. Create new chromosome by mating current chromosomes; apply mutation and recombination as the parent chromosomes mate.
4. Delete members of the population to make room for the new chromosomes.
5. Evaluate the new chromosomes and insert them into the population.
6. If time is up, stop and return the best chromosome; if not, go to 3.

Figure 4.1: Top-level Description of a Genetic Algorithm

4.1.2 The Mathematical Model

Input Model

An organization processes data from n sources of information, i.e. sensors. Each sensor S_i, i = [1, n], can output one letter from its associated alphabet $I_i = \{I_{i1}, I_{i2}, ..., I_{im}\}$. These alphabets describe the basic items of information. The alphabets can include the null element, i.e., the case where no item of information is transmitted. A supersource can be created, which generates events. (Davis, 1991) In Figure 4.2, place I models the supersource, whose *color set* is $I=I_1*I_2*...*I_n$. Transition T1 distributes the information sources from the super source to the appropriate places I_i, i = [1,n]; I_i is the color set for place I_i. Transitions Sensor 1, Sensor 2, ..., Sensor n model the n sensors of the organization, which detect information sources I_1, I_2, ..., I_n, respectively. Place Output 1, Output 2, ..., Output l model the outputs of the organization. They converge through transition T2 into a Sink.

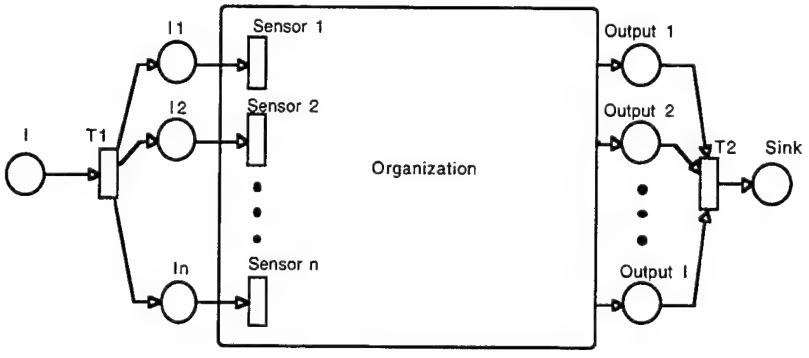


Figure 4.2: Information Sources and Sensors of an Organization

An input situation is represented by an n-dimensional vector $v = (v_1, v_2, ..., v_n)$ where $v_1 \in I_1$, $v_2 \in I_2$, ..., $v_n \in I_n$, i.e. $v \in I_1 * I_2 * ... * I_n$. Therefore, the cross product of the input color sets forms an n-dimensional input space. In this space there is a subspace which includes all *feasible* input situations represented as vectors. We call this subspace *Feasible Input Domain* ***I*** or *Feasible Input Space*. An element in the Feasible Input Domain, represents an *input situation* $v \in \boldsymbol{I}$.

Fixed-Structure Organization

The representation of a fixed-structure organization is based on Petri Net (PN) theory. This section does not describe the Petri Net formalism and the PN-based approaches to generate organization structures; they have been described in earlier reports. The Petri Net representation of the five stage Decision Making Unit (DMU) introduced by Levis (1992) is shown in Figure 4.3. The labels SA, IF, TP, CI and RS are generic names for the *situation assessment*, *information fusion*, *task processing*, *command interpretation*, and *response selection* processes respectively. A DMU receives input or data x from the external environment (sensors). The incoming data are processed in the situation assessment (SA) stage to get the assessed situation z. This variable may be sent to other DMU. If the DMU receives assessed data from other DMU, these data z' are fused together with its own assessment z in the information fusion (IF) stage to get the revised assessed situation z''. The assessed situation is processed further in the task processing (TP) stage to determine the strategy to be used to select a response. The variable v contains both the assessed situation and the strategy to be used in the response selection stage. A particular DMU may receive a command v' from super-ordinate DMU. This is depicted by the use of the command interpretation (CI) stage. The output of that stage is the variable w which contains both the revised situation assessment data and the response selection strategy. Finally, the output or the response of the DMU, y, is generated by the response selection (RS) stage.

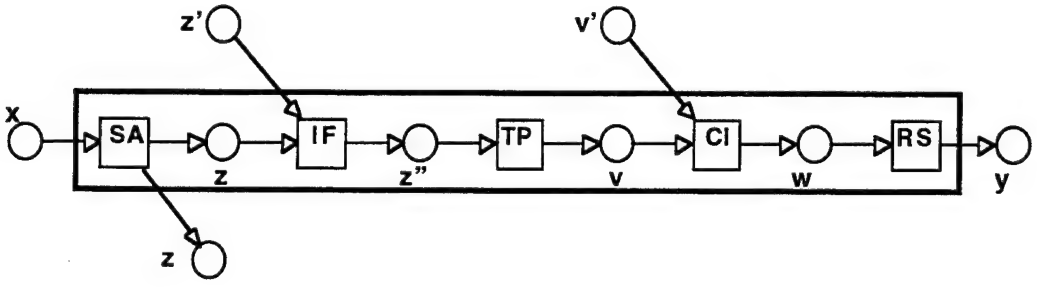


Figure 4.3: Five Stage Model of a DMU

Only certain types of interactions make sense within the model. They are depicted in Figure 4.4. For the sake of clarity, only the links from the ith DMU to the jth DMU are presented. The

symmetrical links from j to i are valid interactions as well. The binary variable e_i represents the *input* to a decision making node. The presence of such a link characterizes the fact that a particular DMU may receive data from the external environment. The binary variable s_i represents the *output* of a decision making node to processes external to the organizational structure considered. The binary variable F_{ij} depicts the transmission of assessed situation from node i to node j; G_{ij} models the transmission of control from the output of a decision making node to the input of another; H_{ij} models the result or processed information sharing type of interaction between two decision making nodes; and C_{ij} represents the flow of instructions or commands from one decision making node to another.

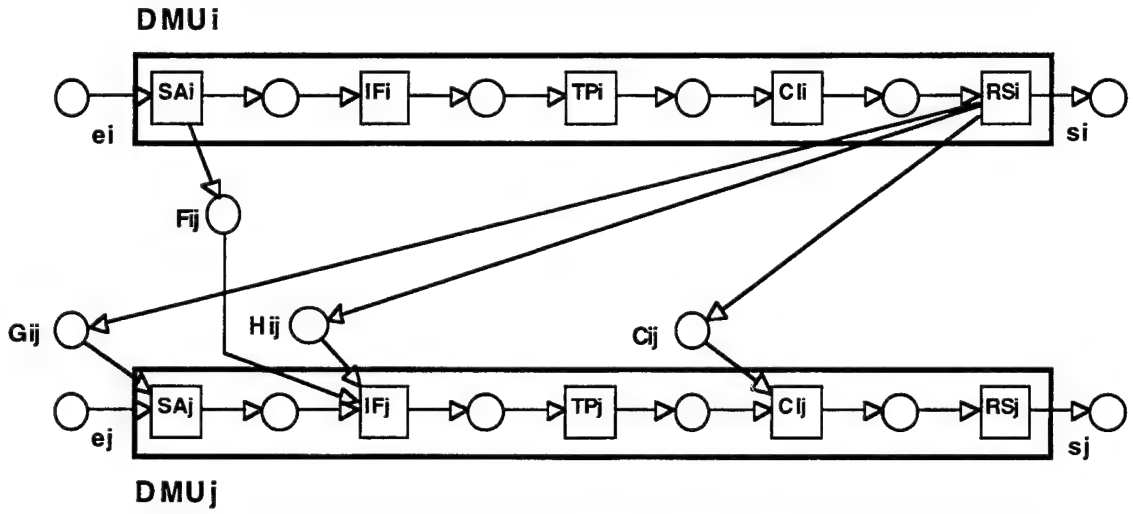


Figure 4.4: Allowable Interactions

The variables e_i, s_i, F_{ij}, G_{ij}, H_{ij}, C_{ij} in Figure 4 are binary variables taking values in {0, 1}, where 1 indicates the presence of the corresponding link in the organizational structure. Note that the value of the variable does not indicate the number of such links which actually exist. The variables are aggregated into two vectors **e** and **s**, and four matrices **F**, **G**, **H**, and **C**. The interaction structure of an organization consisting of n DMUs is, therefore, represented by the tuple:

$\Sigma = \{ \mathbf{e}, \mathbf{s}, \mathbf{F}, \mathbf{G}, \mathbf{H}, \mathbf{C} \}$

where **e** and **s** are n x 1 arrays representing the interactions of the n-DMUs.

$\mathbf{e} = [e_a]$ $\mathbf{s} = [s_a]$ $a = 1,2,...,n$

F, **G**, **H**, and **C** are four n x n arrays representing the interactions among the DMUs of the organizational structure.

$\mathbf{F} = [F_{ab}]$ $\mathbf{G} = [G_{ab}]$ $\mathbf{H} = [H_{ab}]$ $\mathbf{C} = [C_{ab}]$ $b = 1, 2, ..., n$

The diagonal elements of the matrices **F**, **G**, **H**, and **C** are set identically equal to zero; DMUs are not allowed to interact with themselves.

$$F_{aa} = G_{aa} = H_{aa} = C_{aa} = 0 \qquad a = 1, 2, ..., n$$

These relations must hold true for all solutions.

As noted earlier, each fixed organization corresponds to some input situation(s) depicted by the feasible input vector(s), or input situation(s). When dealing with fixed structures only, the associated input situation(s) need not be specified; however, a full description of the input situation(s) is required, together with the corresponding fixed structure, while folding these structure into a variable one.

Example 1

Let an organization is described by the following matrix representation:

$$e = [1 \quad 1] \qquad s = [1 \quad 1]$$

$$\sum_i: \quad F = \begin{bmatrix} \# & 1 \\ 0 & \# \end{bmatrix} \quad G = \begin{bmatrix} \# & 0 \\ 0 & \# \end{bmatrix}$$

$$H = \begin{bmatrix} \# & 0 \\ 0 & \# \end{bmatrix} \quad C = \begin{bmatrix} \# & 0 \\ 0 & \# \end{bmatrix}$$

The Petri net representation of the organization is shown in Figure 4.5.

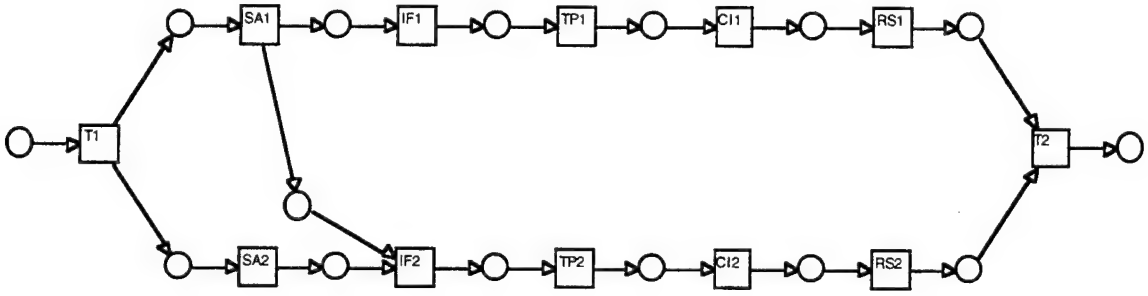


Figure 4.5: PN Representation of a 2-DMU Organization

Variable-Structure Organization

A variable organization can be considered as a set of several fixed-structure organizations associated with every feasible input situation. The components of the variable-structure organization and its structure adapt their functionality and interactions to one of these fixed structures' based on the input currently being processed. In a Colored Petri net (CPN)

representation these several fixed structures, represented as Petri nets, can be folded into a single net. Given this CPN, and an input *token* with *color* v, v$\in I$, the paths which the token follows through the CPN and the task each component performs form a fixed structure. A detailed discussion on CPN representation of variable-structure organizations is presented in Luo and Levis (1992). In this section, we do not adhere to this CPN representation of variable organizations, but use a slightly simpler and static representation which emphasizes the representation of the interactional and functional structures.

Figure 4.6 shows a static representation of a 2-DM variable organization. For the sake of simplicity, assume that all the processing nodes (i.e., SA, IF, TP, CI, and RS) perform identical tasks under different input situations; the only change that occurs from one input to another is of the interactions among these processing nodes. Suppose that the feasible input space $\boldsymbol{I}$ is defined as $\boldsymbol{I}$ = A*B, where A = {a1, a2} and B = {b1, b2} are the input sources(alphabets).

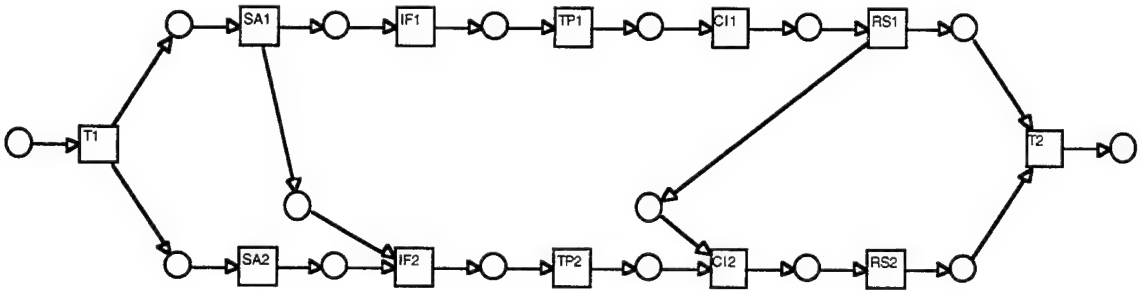


Figure 4.6: A Variable Structure Organization

Structure drawn with solid lines in Figure 4.7 shows a fixed structure, FS1, for input situations (a1, b1) and (a1, b2). When an input (a1, b1) or (a1, b2) is put in input place, it follows the bold path through the organization. The shaded path, while it exists physically, is not utilized when an input (a1, b1) or (a1, b2) is processed by the organization. Similarly fixed-structures for inputs (a2, b1) and (a2, b2), denoted as FS2 and FS3, are shown in Figures 4.8 and 4.9.

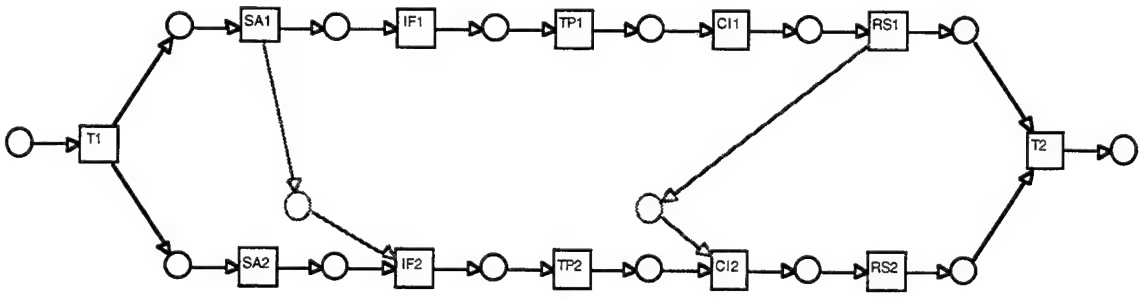


Figure 4.7: Fixed Structure FS1 for (a1, b1) and (a1, b2)

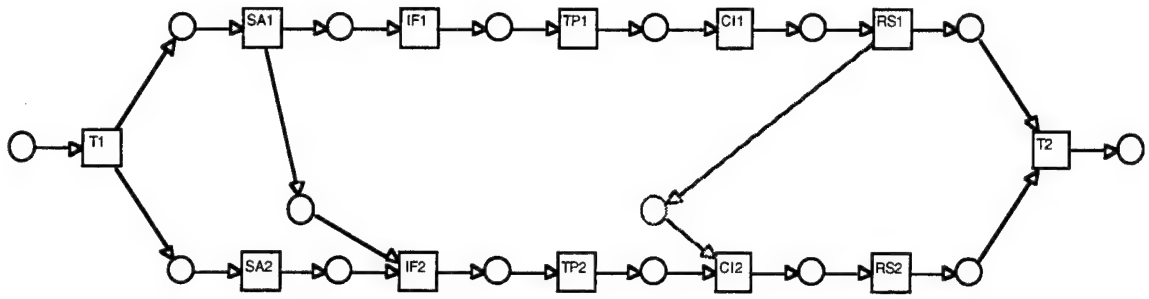


Figure 4.8: Fixed Structure FS2 for (a2, b1)

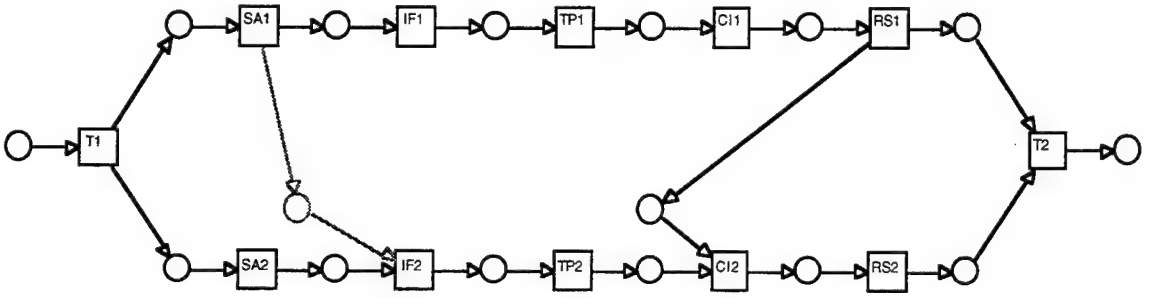


Figure 4.9: Fixed Structure FS3 for (a2, b2)

A variable structure VS is obtained by folding the four fixed structures. VS is denoted as VS= {FS1, FS2, FS3}. A PN representation of VS is constructed by superimposing all fixed-structures; the matrix representation of VS can be obtained by ORing corresponding elements from the matrix representations of constituent fixed structures. However, such a representation is incomplete in the sense that it shows all the possible interactions among the components but hides the variations in these interactions due to different input situations. Once a matrix representation for a VS is obtained by the approach mentioned, an equivalent representation of the VS is constructed for incorporating the variability of the structure. Each link (a place with an input and an output arc) and each transition in the VS is assigned an array of size equal to the cardinality of feasible domain. An element in an array corresponding to a link in VS corresponds to an input situation. The array elements for a link are calculated with the help of a function defined on the feasible input domain, **fi:** I --> {0,1}. **fi**(v) = 0 means that the fixed structure corresponding to input situation v does not contain this link; **fi**(v) = 1 means that the link does exist for the input situation v. A transition/component in VS is also assigned an array, whose elements correspond to input situations. Each element in an array associated with a transition denotes the task performed by the component under the corresponding input situation. Once all the arrays for links and

transitions/components in a VS are specified, the variable structure of the organization is fully defined and represented. Therefore, we can represent the variable structure by an incidence matrix, except that the entries of the matrix are arrays instead of scalars.

The variable structure represented by arrays can be unfolded into a set of fixed structures with each fixed structure corresponding to one input situation. If we fold the three distinct fixed structures described in Figures 4.7 through 4.9, we can obtain a variable structure with its links and transitions annotated by arrays. For example, link from transition SA1 to IF2, denoted as (SA1, IF2), is assigned an array as shown in Figure 4.10. Notice that the only non-zero element in the array corresponds to input situation (a2, b1) representing the fact that the link (SA1, IF2) exists only in FS2.

	b1	b2
a1	0	0
a2	1	0

Figure 4.10: Array for Link from SA1 to IF2

Equivalence of Representations

The previous section has presented two representational mechanisms for variable-structure organizations. In the first approach, a variable organization is represented as a set of several fixed structures, each associated with some input situation(s), with each fixed structure in its matrix representation. The second approach first constructs a matrix representation of the variable organization by ordering the constituent fixed structures and then assigning each link and component in the variable organization a corresponding array. The two approaches are equivalent in the sense that one representation can be constructed uniquely from the other and vice versa. The array elements associated with a link can be determined by the presence or absence of that link in corresponding fixed structures. The matrix representation explicitly identifies the presence and absence of *external* links (links across DMUs) from or to a DMU in an organization. The *internal* links (links between stages of a single DMU) can be determined by the presence or absence of processing stages of a DMU. Figure 4.11 outlines a simple approach of determining the participating stages of a DMU in an organizational structure, and consequently the links internal to the DMU.

Coordination Constraint

The following is a brief and informal description of the algorithm by Luo and Levis (1992) to check coordination constraint for the feasibility of a variable structure.

1.	**If $e_i + G_{ji} > 0$ (for some j) then SAi is present.**
2.	**If $s_i + G_{ij} + H_{ij} + C_{ij} > 0$ (for some j) then RSi is present.**
3.	**If $C_{ji} > 0$ (for some j) then CIi is present.**
4.	**If $F_{ji} + H_{ji} > 0$ (for some j) then IFi is present.**
5.	**If two stages x and y are present then so are all the ones in between the two.**
6.	**All the links connecting the present stages are also present.**

Figure 4.11: Algorithm to Identify the Present Stages of a DMU in an Organization

The algorithm starts with a definition of a Partition Process defined by the user on the input links to each organization member. The process breaks the array associated with the input link into several disjoint partitions. Input situation corresponding to elements in a single partition are indistinguishable to the link or component of a variable-structured organization. Elements from different partitions, however, are distinguishable to the organization. Figure 4.12 describes the user-defined partition processes for the two input components of the organization shown in Figure 4.6. The partitions of the arrays merely states that the component SA1 can distinguish elements from the alphabet A and component SA2 can distinguish elements from alphabet B.

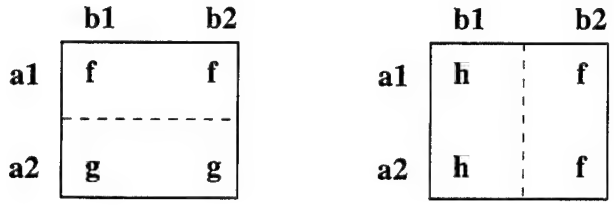
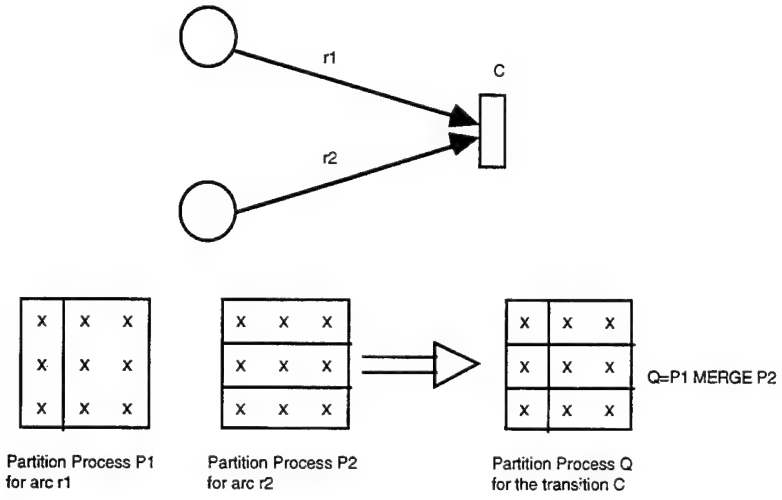


Figure 4.12: Partition Processes of the Arrays Associated with SA1 and SA2

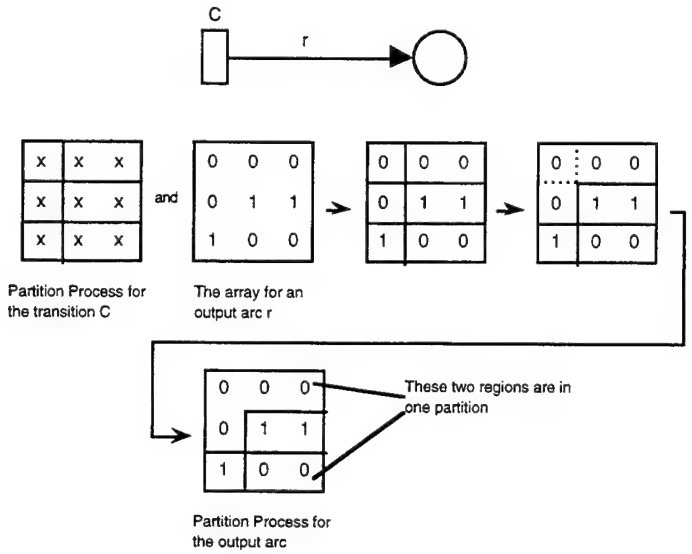
The algorithm also presented an approach to calculate and propagate the partition process for each subsequent link and transition in the folded structure. Figure 4.13 illustrates the method to calculate the partition process for the rest of links and components in the structure.

The coordination constraint described in the previous section indicates that the entries of arrays of output arcs and the component must be the same within a partition of the partition process of an array. The partition processes for the input arcs of a component with multiple inputs are dealt with a different scheme. For components with multiple inputs, the entries of every array of the input arcs must be the same within a partition of all the partition process associated with the input

arcs. The process is repeated for every input arc of a component with multiple inputs. Figure 4.14 shows examples of checking the coordination constraint. The arrays in Figure 4.14 could be of an arc or component. The solid lines partition each array into six partitions. The entries in each partition of the first array are the same; consequently, the coordination constraint is satisfied. In the second case, one of the partitions has two different elements. There the coordination constraint is not satisfied. Therefore, if we can find the partition processes for all the components in the organization, and find that they meet the coordination constraint, a realizable or feasible variable structure is obtained.



(a) Derivation of the Partition Process for a Component



(b) The Partition for an Output Arc

Figure 4.13: Derivation of Partition Processes

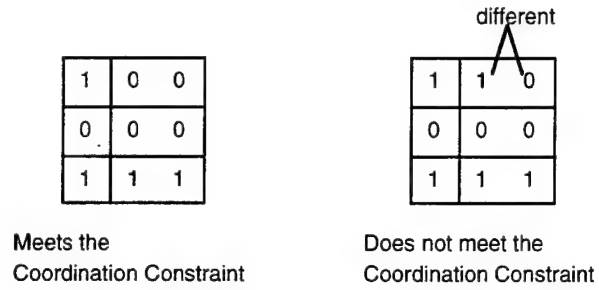


Figure 4.14: The Coordination Constraint

Figure 4.15 describes the procedure to find feasible variable structures. At the first stage, variable structures are constructed by arbitrarily choosing a fixed structure from each set of an input situation. At the second stage, an algorithm is used to check the coordination constraint for each generated variable structure. Only the variable structure which meet the coordination constraint are retained and stored as feasible variable structures. The coordination constraint is like a filter which rules out the unrealizable variable structures.

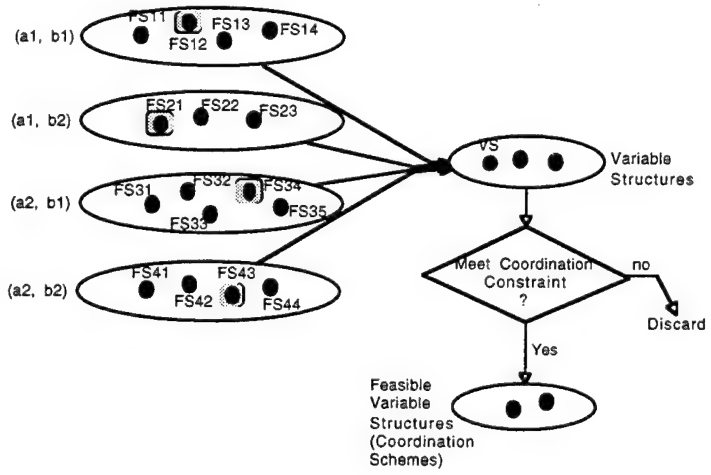


Figure 4.15: Design Procedure For Feasible Variable Structures

4.1.3 Generation of Structures By GA

The design procedure in Figure 4.15 is carried out in two stages: 1) sets of feasible fixed structures are generated for each input situation, 2) One element from each of these sets of fixed structures is selected arbitrarily and a variable organization is formed by folding all the selected fixed structures. Finally, the variable structure is checked for coordination constraint. The process is computationally expensive and *blind* at both these stages; first all the feasible fixed structures are

generated independent of the others with no information exchange among the processes to determine which interactional structures *might* result in feasible folding, and then the structures are *arbitrarily* selected from each set to form a variable structure. It is this reason that led Zaidi and Levis (1996) to investigate the genetic algorithm to generate fixed structures. The genetic algorithm performs a parallel heuristic search of the solution space for feasible solutions. The same approach is extended for the generation of variable-structure organization. The rest of the section describes the approach and identify issues that are required to be investigated.

Encoding

The encoding of a fixed organizational structure into a chromosome converts the matrix representation $\Sigma = \{ \mathbf{e}, \mathbf{s}, \mathbf{F}, \mathbf{G}, \mathbf{H}, \mathbf{C} \}$ into a bit string, termed chromosome, as illustrated in the following example.

Example 2

Let the organizational structure of an organization is given by the tuple Σ.

$$\Sigma: \quad \mathbf{e} = [0 \quad 1] \qquad \mathbf{s} = [1 \quad 1],$$

$$\mathbf{F} = \begin{bmatrix} \# & 0 \\ 1 & \# \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} \# & 0 \\ 1 & \# \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} \# & 0 \\ 0 & \# \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} \# & 1 \\ 0 & \# \end{bmatrix}$$

The bit string representing the chromosome, Σ, is obtained by the following encoding:

$$\Sigma: \quad \overbrace{0\ 1}^{e} \ \overbrace{0\ 1}^{F} \ \overbrace{0\ 1}^{G} \ \overbrace{0\ 0}^{H} \ \overbrace{1\ 0}^{C} \ \overbrace{1\ 1}^{s}$$

The diagonal elements of the matrices **F**, **G**, **H**, and **C** are ignored in the bit string representation since they remain zero throughout the design procedure.

An i^{th} bit in the bit string representation of an organizational structure, Σ, is accessed through the notation $\Sigma[I]$, i.e., $\Sigma[4] = 1$ in Example 2.

The length of the string representing an organizational structure Σ is denoted by $|\Sigma|$, i.e., $|\Sigma| = 12$ in Example 2. The length of the bit string (chromosome) representing an organizational structure (individual in a population) with n DMUs is given by:

$|\Sigma| = 4n^2 - 2n$ where n is the number of DMUs in Σ

Therefore, the index 'i' in $\Sigma[i]$ takes on the values; $1 \le i \le 4n^2 - 2n$.

A variable organization is represented as an organism with layers of chromosomes each representing a fixed structure along with its associated input situation. Figure 4.16 represents an encoding of the variable structure VS of Figure 4.6.

```
1 1 0 0 0 0 0 0 0 0 1 1 [for input situation (a1, b1)]
1 1 0 0 0 0 0 0 0 0 1 1 [for input situation (a1, b2)]
1 1 1 0 0 0 0 0 0 0 1 1 [for input situation (a2 , b1)]
1 1 0 0 0 0 0 0 1 0 1 1 [for input situation (a2, b2)]
```

Figure 4.16: Organism Representing a Variable Structure

Design Requirements

A designer of an organizational structure is first required to specify the feasible input domain for the organization to be built. Therefore, the first requirement for the design procedure is to model in input source. Once the feasible input domain is specified, the designer is asked to input the interactional requirements for every fixed structure corresponding to a (some) input situation(s). The interactional requirements for a fixed structure participating in a variable organization are provided by the designer in terms of user-defined constraints, Ru. Requirements for each participating fixed structure are input to the design algorithm by putting 1's and 0's at corresponding places in the matrix representation. The user-defined constraints are taken for every input vector in the input domain.

The user-defined constraints for a participating fixed organization, in terms of its constituent DMUs, are given as the tuple Σ_i.

$$e = [1 \quad x] \qquad s = [0 \quad x]$$

$$\Sigma_i: \quad F = \begin{bmatrix} \# & 1 \\ x & \# \end{bmatrix} \quad G = \begin{bmatrix} \# & 0 \\ 0 & \# \end{bmatrix}$$

$$H = \begin{bmatrix} \# & x \\ x & \# \end{bmatrix} \quad C = \begin{bmatrix} \# & 0 \\ 0 & \# \end{bmatrix}$$

The bit string representation of Σ_i is given as:

$$\Sigma_i: \quad 1 \; x \; 1 \; x \; 0 \; 0 \; x \; x \; 0 \; 0 \; 0 \; x$$

The x's in the arrays represent the unspecified elements or optional links. The optional links determine the degree of freedom left in the design process, and potentially yield a number of feasible fixed structures for the corresponding input situation.

Initial Population

An organism with each constituent chromosome representing user-defined constraints, shows the building block or *schema* for the generation of future populations of structures. A 1 or 0 at any position means that the chromosomes in future populations must have the same value at that position for them to belong to the schema. The x's represent the genes (interactions) that can be replaced by either 1's or 0's genetically to generate new populations of solutions.

The first step in the genetic algorithm approach requires an initial population of organism to start the process. In the present approach, the bit strings representing the Universal and the Kernel Nets corresponding to individual chromosomes are used to initialize the population; an initial population of variable structures, or organism in the genetic terminology, contains two individuals, where the one individual contains all the chromosomes in their Universal net representation and the other contains chromosomes in their Kernel net form. This way we start the genetic search at one end from the maximally connected organization and at the other end from the least connected organization.

Structural and Coordination Requirements

In order to have a feasible variable organization, or a fit organism in genetic terminology, each fixed structure, or chromosome in the organism, is required to be a feasible organization both in terms of structural requirements and user-defined constraints. Once we have all chromosomes in an organism represent feasible fixed organization, the organism is checked for coordination constraint. The report has already described the coordination constraints. In the present approach, each organism is converted to its array representation and partition processes are calculated for every component and interactional link in the variable organization, and partitions are checked for feasibility.

A brief description of structural constraints for individual fixed organization follows:

Structural Constraints

(R1) The Ordinary Petri net that corresponds to a fixed structure should be connected, i.e., there should be at least one (undirected) path between any two nodes in the net. A directed path should exist from the source place to every node of the net and from every node to the sink.

The genetic implementation of R1 checks each chromosome in an organism to establish the internal structure of each constituent DMU. For a connected organizational architecture the internal structures of all DMUs must fall within the following four possibilities.

- SA alone with y = z
- SA, IF, TP, CI, and RS
- IF, TP, CI, and RS with x = z'
- CI and RS with x = v'

In addition to checking the internal structures for DMUs, the following checks are also performed to ensure that each fixed organization is also connected to the external environment through inputs and outputs (sink and source places.)

$$\sum_{j=1}^{n} \Sigma[j] \geq 1$$

$$\sum_{j=1}^{n} \Sigma[4n^2 - 3n + j] \geq 1$$

(R2) The Ordinary Petri net that corresponds to a fixed structure should have no loops. i.e., the structure must be acyclic. An algorithm has been presented in [1] that checks for this constraint in every chromosome of an organism.

(R3) In the Ordinary Petri net that corresponds to a fixed structure, there can be at most one link from the RS stage of a DMU i to another DMU j, i.e., for each i and j, only one element of the triplet $\{G_{ij}, H_{ij}, C_{ij}\}$ can be non-zero. The analytical expression of this constraint is given as:

$$\forall(i,j) \quad G_{ij} + H_{ij} + C_{ij} \leq 1 \qquad i \neq j;\ i, j = 1, 2, ..., n$$

The analytical expression of this constraint, applied to the genetic representation of a fixed structure, is given as:

$$\forall i \quad \Sigma[n^2 + i] + \Sigma[2n^2 - n + i] + \Sigma[3n^2 - 2n + i] \leq 1 \qquad i = 1, 2, ..., (n^2 - n)$$

(R4) Information fusion can take place only at the IF and CI stages. Consequently, the SA stage of a DMU can either receive information from the external environment, or an output from another DMU. The translation of this constraint into mathematical terms follows:

$$\forall j \quad e_j + \sum_{i=1}^{n} G_{ij} \leq 1 \qquad j = 1, 2, ..., n$$

The translation of this constraint for the genetic representation of a fixed structure follows:

$$\forall j \quad \Sigma[j] + \sum_{\substack{i=1 \\ i \neq j}}^{n} \Sigma[k_{i,j}] \leq 1 \qquad j = 1, 2, ..., n$$

where

$$k = n^2 + (i-1)(n-1) + j - 1 \quad \text{for} \quad i < j$$
$$k = n^2 + (i-1)(n-1) + j \quad \text{for} \quad i > j$$

Computation of Solutions

In the present implementation of the genetic algorithm, an organism consists of several chromosomes each representing a constituent fixed organization. The requirements for one fixed organization are independent of the requirements of the other. Therefore, each chromosome in an organism is allowed to enhance its population independent of the rest. The approach used to generate new populations of chromosomes, the genetic algorithm proposed by Zaidi and Levis (1996), is applied to corresponding chromosomes of a population of organisms. However, when the chromosomes of an organism are folded to form a variable structure, the organism is checked for the coordination constraint. The information obtained during the checking of coordination constraint is used to tailor the genetic operators. Each member (organism) of a population is evaluated against a fitness function. The evaluation results in an assignment of a fitness value to each and every member of the population.

Evaluation Function

The evaluation function, used in the methodology, evaluates an organism in a population on the basis of three parameters, and assigns a fitness value accordingly. The fitness value assigned to an individual is calculated by the following formula.

$$Fitness = f(x, y, z)$$

where

x represents a parameter whose value is determined by some user-defined criterion, i.e., number of interactional links.

y represents the parameter whose value depends upon the structural feasibility of individual fixed structures in the organism.

z represents a parameter whose value depends upon the feasibility of coordination structure of the organism, determined by the coordination constraint.

The fitness assigned to organism determines the 'strong' and 'weak' individuals in a population. The stronger elements are retained in the population and allowed to reproduce and generate following populations of organisms. Figure 4.17 and Figure 4.18 presents illustrations of the two genetic operator, crossover and mutation, used in the approach. The two operators are applied to chromosomes of organisms. The coordination structure of a *weak* organism provides

useful information regarding the problematic interactions —links where the coordination constraint is violated. This information is used to determine the mutation rate or crossover point for the genetic operators used in the generation of subsequent generations. For example, during evaluation of the coordination constraint if a partition (associated with a link) is found with different elements in it. The mutation rate for that particular link in a fixed structure, which contributes to this feasibility, is increased to eliminate this link in the next generation. In the present approach, we are investigating the effect of an increased mutation rate on the design procedure. The same information, however, can be used to determine the crossover point for the mating operation performed on the present generation.

Parent 1:	1	1	1	1	0	0	1	1	0	0	0	1
Parent 2:	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Child 1:	1	**0**	1	**0**	0	**0**	1	**0**	0	**0**	0	**0**
Child 2:	**1**	1	**1**	1	**0**	0	**0**	1	**0**	0	**0**	1

Figure 4.17a : Crossover

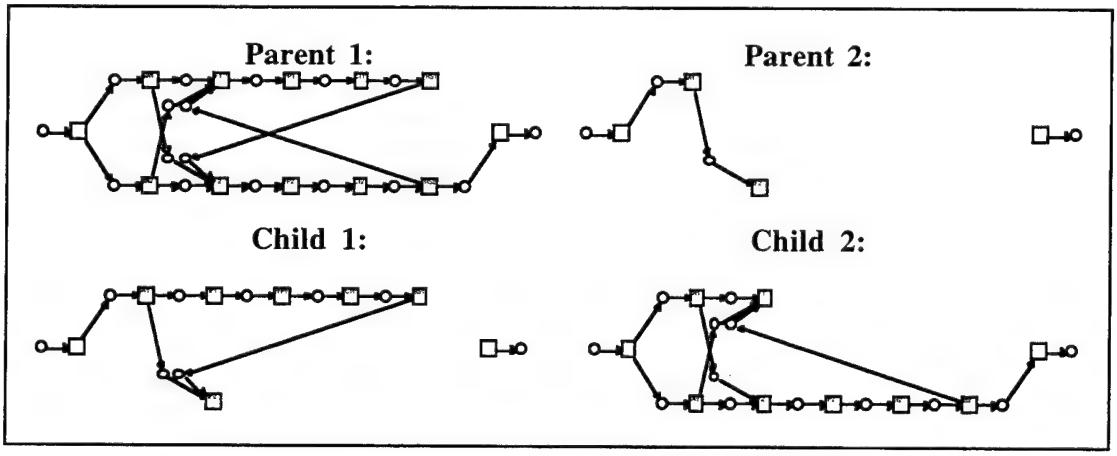


Figure 4.17b: Petri Net Representation of Crossover

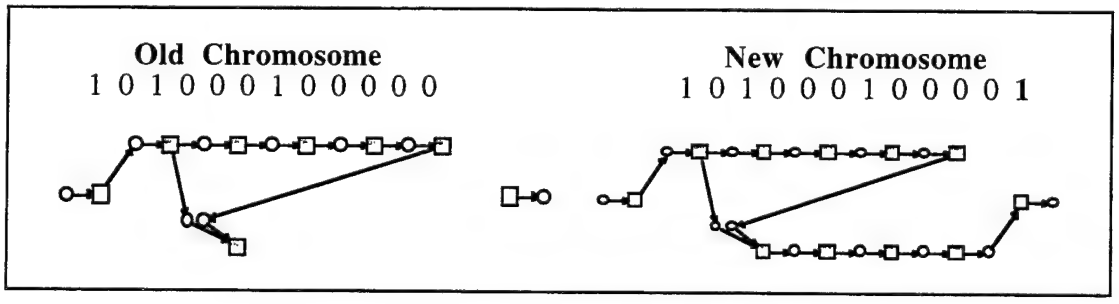


Figure 4.18: An Example of Bit Mutation

4.1.4 Conclusion

The report presents results of an ongoing effort on generating variable structure organizations by Genetic algorithms. The genetic approach replaces the previous approach of constructing variable organizations. The previous approach generates feasible structures for all participating fixed organizations, and then folds these structures by arbitrarily selecting them. As mentioned earlier, that the approach is computationally intensive since it generates all sets of feasible fixed structures first and then combinatorially evaluates these structures for coordination feasibility. The new approach using genetic algorithms generates fixed structures in an incremental fashion, and checks them for coordination feasibility right away. The information obtained during the evaluation of the coordination constraint is used to direct the search for other feasible fixed structures. This information not only is useful in narrowing down the search for other feasible structures, but is also helpful in eliminating a potentially large number of feasible structures with infeasible coordination structure while folded with others.

The effort currently focuses on ways to improve the genetic operators used in the methodology to yield better and earlier results. The present approach lacks a mechanism for checking the user input for infeasibility prior to initiating the design procedure. A comparative study of the two design approaches based on an application is in progress. Work is also being done on a user-friendly implementation of the design methodology.

REFERENCES

Davis, L., (1991) *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold, 1991.

Deamël, J. J., and A. H. Levis, (1994) "On Generating Variable Structure Architectures for Decision Making Systems," *Information and Decision Technologies,* Vol. 19, pp. 233-255.

Levis, A. H., "A Colored Petri Net Model of Intelligent Nodes," in *Robotics and Flexible Manufacturing Systems*, J. C. Gentina and S. G. Tzafestas, Eds., The Netherlands: Elsevier Science Publishers B. V., 1992.

Luo, Z., and A. H. Levis, "Coordination in Distributed Intelligence Systems," *Proc. 1992 IEEE International Conference on Systems, Man, and Cybernetics*, Chicago, Illinois, October 18-21, 1992 pp. 668-673. New York: IEEE Press.

Zaidi, A. K. and A. H. Levis, (1966) "On Generating DIS Architectures Using Genetic Algorithms," *Proc. International Symposium on Command and Control Research and Technology*, Monterey, California, June-July, 1996.

4.2 ADAPTIVE ARCHITECTURES FOR COMMAND AND CONTROL: PRE-EXPERIMENTAL, EXECUTABLE MODELS (Handley, Z. Zaidi, Levis)

4.2.1 Introduction

The Adaptive Architectures for Command and Control (A2C2) Program is a research initiative to investigate processes of adaptation in Joint Command and Control architectures at the Commander Joint Task Force level. GMU's tasks in this program were to develop executable models of proposed architectures and exercise them in a simulation mode in order to study their dynamic behavior. The models that were created are based on an Object Oriented design approach which allows the same model to be re-configured to different architectures simply by changing the Decision Maker - Platform - Task mapping information. The models were then implemented using Colored Petri nets (CPN) to create discrete event models which can be simulated. The models are exercised by a scenario comprised of the 175 tasks expected in the DDD (Distribute Dynamic Decision-making) experimental environment. The modeling and results analysis was done before the experiments that include human participants were carried out in order to support (a) the formulation of hypotheses; (b) the determination of key variables and parameter values; and (c) the prediction of organizational performance and processes of adaptation. After the experiments are run, the model-generated data will be used as a baseline to compare and analyze the experimental data.

4.2.2 Objectives

A main objective of the A2C2 Research Plan is to strengthen the interactions between specific research teams involved in the program. GMU worked with both the University of Connecticut (UConn) and the Naval Postgraduate School (NPS) to coordinate approaches to architectural modeling during the pre-experimental modeling stage. GMU created models that represented the architectural forms designed by UConn. The results of GMU's modeling were fed back to UConn in an iterative design process to allow for possible modification of the architectures based on the model results. The models were executed with scenario information from NPS and pre-experimental parameter variation results were passed back to NPS to use in the final design of the experiment. All three teams tried to make as few assumptions as possible outside of the experimental specifications or architectural designs in order to keep consistency across the models and design processes and to produce valid and comparable results.

GMU's specific program objective was to create pre-experimental, executable models. The architectures have been designed and modeled in order to respond to a specific Joint Task Force (JTF) mission: to secure the port and airfield of a friendly nation that has experienced a hostile take over. The trigger of adaptation in this mission is a reduction in assets: different architectures have been designed and modeled to represent the mission under full and reduced number of platforms. UConn designed different feasible architectures by using aggregated task functions and static measures of organizational performance. GMU used this information to reconfigure the model to the specific architectures. NPS provided information in the form of a written descriptive scenario, a task graph, and DDD parameter information. This information was used to design the scenario driver for exercising the models over the full range of inputs and for carrying out behavior and performance analysis. Once the baseline behavior of the architecture had been observed, the architectures were exercised over the full modeling space in order to determine which parameters should be varied over what ranges and what variables should be held constant or measured. The (sometime unexpected) behavior of the model as it was exercised can be used to adjust the hypotheses that will be tested experimentally. This third experiment in the A2C2 program is the first time pre-experimental modeling was completed and a preview of the experimental outcomes provided to the experimental designers prior to the conduct of the experiment.

4.2.3 Model Design

In the two previous iterations of the A2C2 program, GMU used the Structured Analysis Design Technique (SADT) to create the executable models. This approach uses the decomposition of the scenario into tasks and functions and the data exchange between them as the foundation of the model. However, as its name implies, it is very structured and any modification to a part of the modeled system, for example, the organizational architecture, requires a complete restructuring of the model. In order to accommodate the possibility of any number of architectures, and iterative revisions, it was necessary to create a model that would be easily re-configurable. Also in this iteration the sea battle was included in the model, which was not done previously. This consists of a series of independent tasks occurring over time. These are difficult to represent in the SADT because they are independent of the other activities of the system. By taking an Object Oriented (OO) approach, a concept borrowed from software engineering, a much more flexible model was created.

In order to create an OO based model, the structure of the system is defined initially instead of the process. By examining the basic characteristics of the system, three abstract entities or object classes were defined: Decision Maker (DM), Platform, and Task. These are general definitions that represent the all the object classes as shown in Figure 4.19. Different instances of a particular

object are generated by defining the attributes specific to that instance. For example, to instantiate the platform instance whose ID is SAT, the fixed attributes would be set as follows: Resources: 6 IDES, Owner Platform = CV, Trigger Eliminated = No, and Velocity = 0.7. Its initial conditions would fix the variable attributes until they are changed in order to accomplish a particular task.

DM	PLATFORM	TASK
ID:DM_ID	**ID**:PLT_ID	**ID**:TSK_ID
Attributes (Fixed):	**Attributes** (Fixed):	**Attributes** (Fixed)
Skills	Resources	Value
Owned Platforms	Owner Platform	Time
Assigned Tasks	Trigger Eliminated?	Task Type
	Velocity	Function Type
		Location
		Resource Requirements
		Skill Requirements
		Multiplicity
	Attributes (Variable)	
	Current Location	
	Distance to Travel	
Methods:	**Methods:**	**Methods:**
Assess Platform	Travels to Task	Execute Task
Assess Request		
Enable Task		
Launch Platform		
Messages: Request Platform Platform OK Launch Platform		

Figure 4.19: Object Definitions

The interactions of instantiations of the object classes is how the system accomplishes tasks and completes the mission objectives. Three views of the system are necessary to define the model. The first view, the Object View, describes the relationships between the objects. This is represented by the Entity-Relationship (ER) Diagram shown in Figure 4.20.

The second view, the functional view, must be defined for each method in the object definitions. For example, Figure 4.21 show the functional view for the task method Execute Task. Central to the diagram is the method, the arcs show the necessary data, and the boxes show the objects providing that data.

Finally the dynamic view depicts the behavioral aspects of the model. An example of a dynamic view is shown in Figure 4.22. This shows two instances of a DM interacting to launch two platforms to accomplish a task.

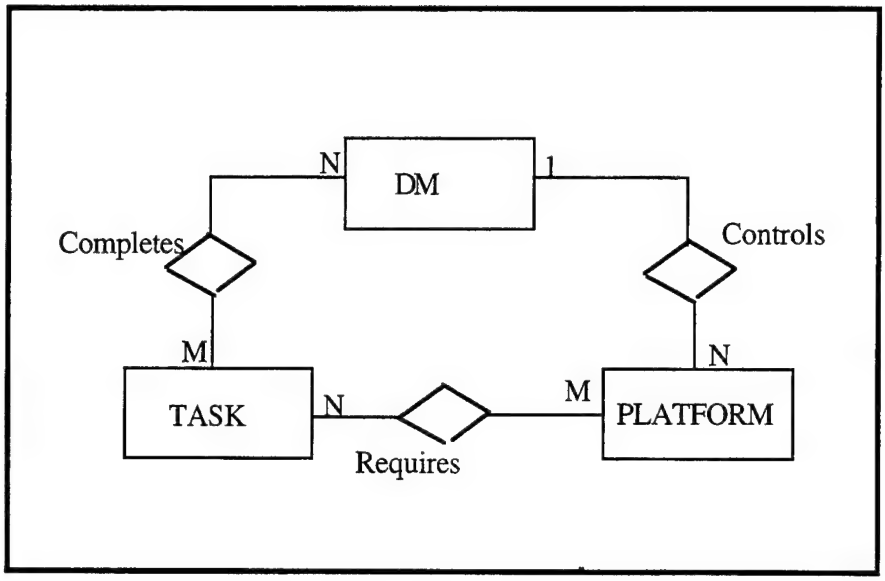


Figure 4.20: Object View

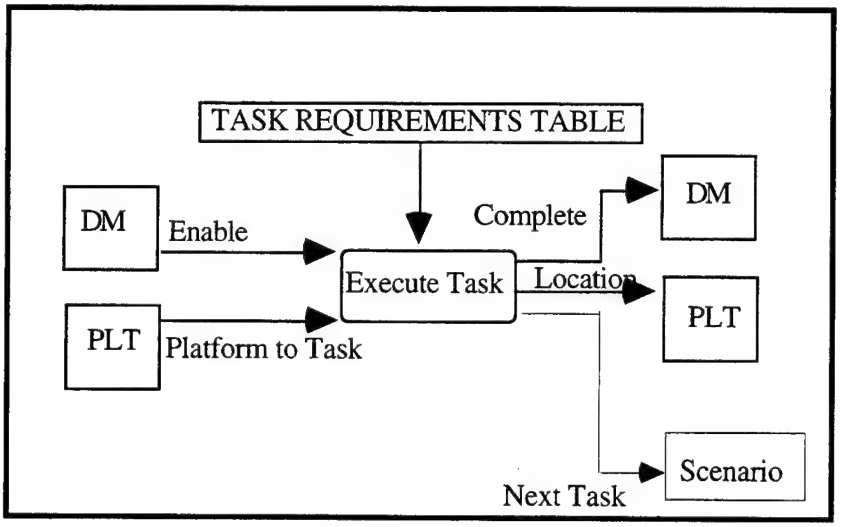


Figure 4.21: Functional View

Once the model has been completed designed, it can be implemented using Colored Petri nets (CPN) in order to develop an executable model. The information from the three static views above is incorporated in one model. A simplified view of the top level of the model is shown below in Figure 4.23. One CPN model was developed and then customized for each instantiation of the architectures by changing the initial marking which contains the mapping between DMs, Platforms and Tasks. The structure of the CPN model follows the object view: The top level shows the

interactions between the three objects and the subpages each describe the specific instances of the objects.

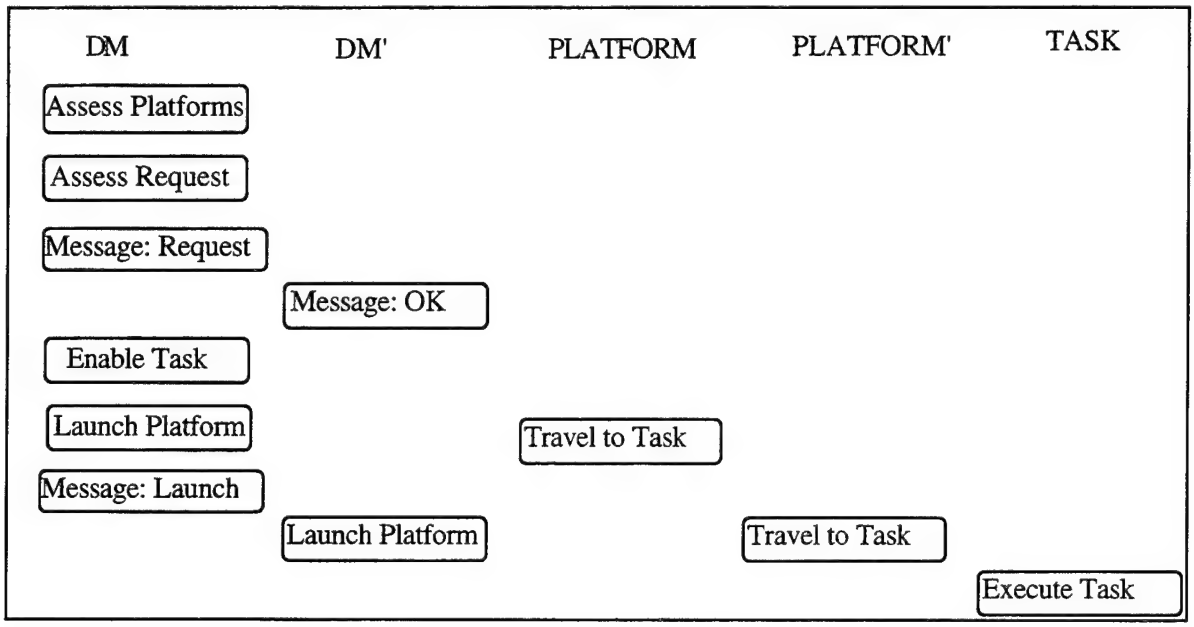


Figure 4.22: Dynamic View

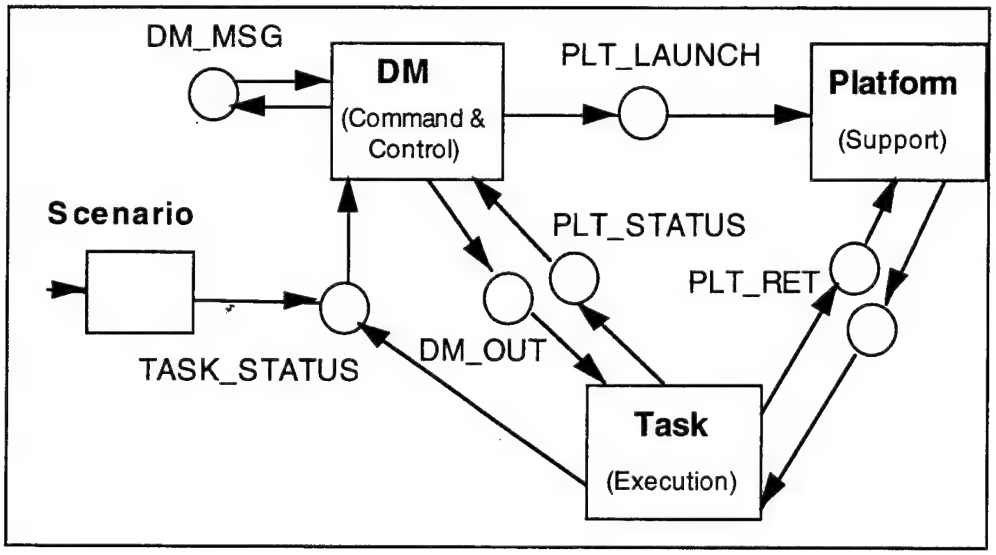


Figure 4.23: Simplified Top Level CPN Model

This simplified view of the model shows the three interacting object classes and the scenario driver. It illustrates the basic premise of the model. When a task occurs, it is recognized by the DM mapped to that task. The DM launches the appropriate platform mapped to that task. Travel time to

the task is included in the model based on the individual platform's travel times and its previous location. Once the platform is available and enabled by the DM, the task executes for the task duration time defined for that task. The platform is then available to be assigned to another task. In the case of coordinated tasks, the DMs send messages over communication channels to all DMs mapped to the task to align the necessary platforms before enabling the task.

4.2.4 Incorporating the Architectures

The architectures implemented with the model were designed by UConn. During the course of the pre-experimental modeling, GMU had sufficient information to model 5 architectures: A0-6, A1-6, A1-4, A2-5, A2-6. Only three of these architectures will be included in the final experiment: A0-6, A1-4, A2-5, therefore these will be the only ones discussed in the rest of this report. A0-6 is the pre-trigger architecture, it has 6 DMs and 36 active platforms. A1 and A2 are post trigger architectures. A1 has 4 DMs and 20 active platforms and A2 has 5 DMs and 20 active platforms. A2 was designed to be "closer" to A0, but not do as well in performance, while A1 was more "distant" but would do better under reduced resources.

The UConn team provided architectural information in the following form: an organizational chart showing the number of DMs and their command hierarchy, the platforms assigned to each DM, and a Gantt chart with accompanying parameter file that depicted the scheduling of resources to *aggregated* tasks for a period of time. The GMU model could not use the information in this form; the platform to *individual* task mapping needed to be extrapolated from this information. In order to do their design process, UConn aggregates tasks instead of using individual tasks in their modeling process. Each object on the NPS task chart, as shown in Figure 4.24, becomes an aggregated task: the characteristic task requirement vectors are summed together and platforms are assigned to complete this aggregated task in one time block. The UConn team uses 24 aggregated tasks in its design process. In order to "undo" this process and obtain the individual task mappings, the original task requirement vectors from the DDD parameter file are examined and an appropriate platform is picked from the group of platforms that was assigned to that aggregated task group. Note that in the scenario driver used to simulate the model, individual tasks may occur outside of the window of the aggregated task used in the UConn Gantt chart . For example, the UConn aggregated tasks for the North Beach were related to a Task - Platform mapping, and by including the DM - Platform mapping provided, a complete mapping for the model as shown in Figure 4.25 was obtained.

UConn Tasks	**[210] North Beach Encounter**		*A0 - 6*
211	[T9-PZ2] (E:SC: Mines) [x2]	SMC -42	DM2
241	[T19-PZ2] (D:SC: Neutral Patrol Boat) [x1]	TARP -27	DM0
211	[T16-PZ2] (D:SC: Patrol Boat) [x1]	DDG - 6	DM2
212	[T20-PZ2A] (MM:MD: Remove Wounded)	MED -41	DM2
	[220] North Beach Mission		
220	[T4-PZ2] (MT:GM: Take Beach)	CAS-26 + INF-50 + CAS-25	
			DM4/2/1
	[230] North Beach Defend		
232	[T20-PZ2B] (MM:MD: Remove Wounded)	MED-31	DM2
231	[T5-PZ2] (D:GC: Artillery) [x4]	AH - 40	DM4
231	[T6-PZ2] (D:GC:Frog) [x5]	VF-18	DM0
231	[T11-PZ2] (D:AC: Air-Ground) [x1]	CAS-24	DM4

Figure 4.24: Task-Resource-DM Mapping for Full Resource Architecture

UConn Task	**[210] North Beach Encounter**		*A2-5*	*A1-4*
211	[T9-PZ2] (E:SC: Mines) [x2]	SMC -16	DM1	DM1
241	[T19-PZ2] (D:SC: Neutral Patrol Boat) [x1]	SAT -8	DM0	DM0
211	[T16-PZ2] (D:SC: Patrol Boat) [x1]	CAS -13	DM0	DM1
212	[T20-PZ2A] (MM:MD: Remove Wounded)	MED -15	DM3	DM5
	[220] North Beach Mission			
220	[T4-PZ2] (MT:GM: Take Beach)	CAS-12 + INF-28 + INF-27		
			DM2/2/3	DM2/4/3
	[230] North Beach Defend			
232	[T20-PZ2B] (MM:MD: Remove Wounded)	MED-15	DM3	DM5
231	[T5-PZ2] (D:GC: Artillery) [x4]	CAS-14	DM0	DM5
231	[T6-PZ2] (D:GC:Frog) [x5]	VF-10	DM4	DM2
231	[T11-PZ2] (D:AC: Air-Ground) [x1]	CAS-14	DM0	DM5

Figure 4.25: Task-Resource-DM Mapping for Reduced Resource Architectures

The mapping for A0 has the same tasks, but with different assigned resources since it is before the trigger as shown in Figure 4.24.

4.2.5 The Scenario Driver

An important aspect of the GMU model is the scenario driver that drives the execution of the models. The scenario needs to represent the type and tempo of inputs that will be realized in the DDD simulator in the actual experiment. NPS provided a Task Graph, as shown in Figure 4.26. This Task Graph represents a partial ordering of the inputs that will be provided. The circles represent the *mission* tasks. These are the objectives that team must achieve. Before each mission task may be some *encounter* tasks; these represent threats or obstacles that must be cleared before the mission task is obtainable. After the mission tasks, and throughout the course of the scenario at sea, are *defend* tasks. These are possible threats that may occur and should be destroyed to protect friendly assets and/or territory already under the control of friendly forces.

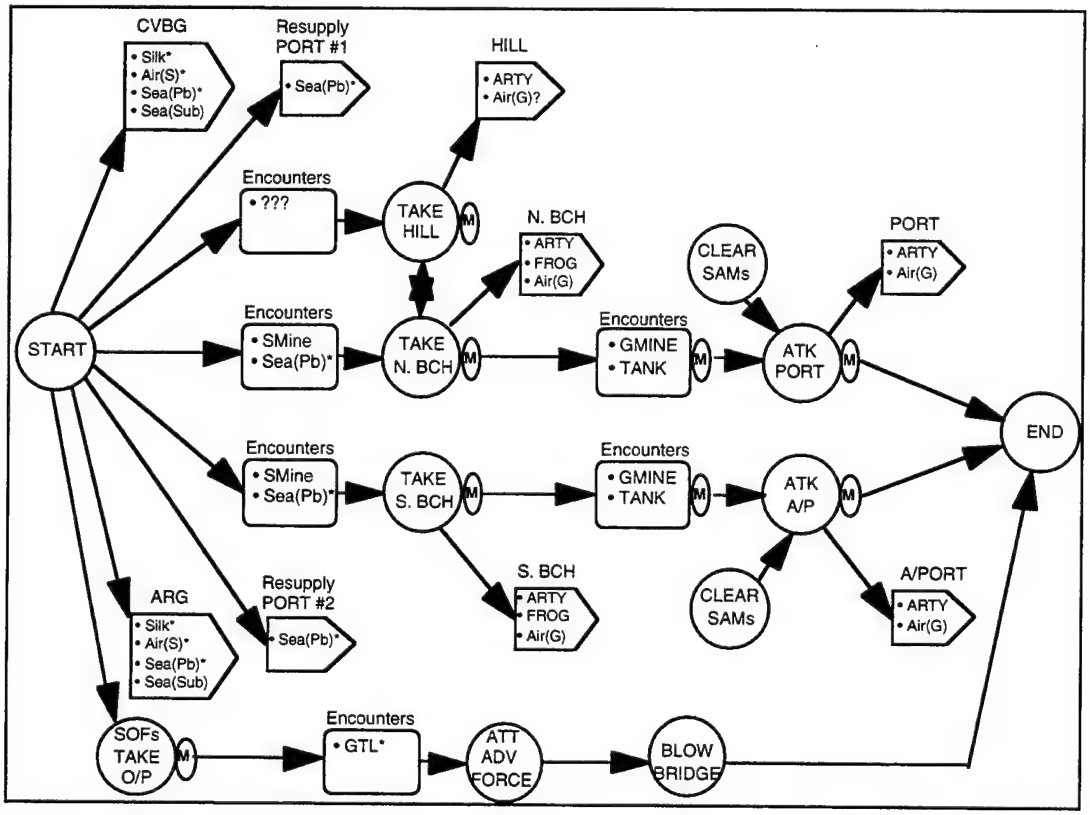


Figure 4.26: NPS Task Graph

Although general task precedence is shown in Figure 4.26, the component individual tasks and their frequency are only given in the preliminary DDD parameter file, also provided by NPS, as shown in Figure 4.27.

ID	TASK	No.	TYPE	DURATION	COMMENTS
0	HILLS	1	M	10	
1	AIRPORT	1	M	30	
2	SEAPORT	1	M	30	
3	HOLD	3	M	90	
4	BEACH	2	M	10	
5	ARTY	20	D	10	Target PZs 0-4
6	FROG	10	D	10	Target NS Beaches
7	SILK(H)	6	D	15	2 North, 4 South
8	GMINE	4	E	20	2 on each Road
9	SMINE	8	E	20	2 at each Beach, 4 Drifting Sea
10	AIR(Sea)	12	D	20	6 each North and South
11	AIR(Gnd)	3+	D	20	Target PZs 0, 2-4
12	SAM(H)	3	E	20	2 at Port, 1 at Airfield
13	SAM(N)	3	E	20	2 at Port, 1 at Airfield
14	TANK(H)	5	E	10	2 each Road, 1 at Port
15	SILK(N)	10	D	15	6 North, 4 South
16	SEA(Pb)	8	D	10	2 each PZs 5-8
17	SEA(Sub)	6	D	10	3 each North and South
18	SEA(N)	20?	D	10	
19	SEA(NPb)	16	D	10	
20	MEDVC	6	M	60	
21	AIR(N)	16	D	20	
22	MISSILE	??			
23	DUMMY				
24	GTL(H)	1	E	10	
25	GTN(N)	6	E	10	3 on each Bridge Approach
26	OPOST	1	M	10	
27	BRIDGE	1	M	20	

Figure 4.27: DDD Task Parameter File

GMU combined the task precedence information from the task graph along with the occurrence information from the parametric file to produce the input scenario. Note that the task

duration is given in this file but there are no other time anchors. The input scenario schedules 175 individual tasks for the model to execute: some tasks occur at multiple times at different intervals throughout the scenario, and some tasks do not appear until later in the scenario as depicted in the task graph.

4.2.6 Results

Behavioral Modeling Results

By executing the model with the baseline scenario, a dynamic view of the parameters used to design the architectures was provided for behavioral analysis. With the baseline scenario all tasks were completed with some latency; times corresponded roughly to UConn Gantt Chart times. The architectures were compared based on their completion time of the mission tasks as shown in Figure 4.28. Note that all times are in simulation seconds.

	A0-6	A1-4	A2-5
Completion Times: Mission Tasks:			
HILL (T0-PZ1)	464	514	514
NORTH BEACH (T4-PZ2)	407	447	447
SOUTH BEACH (T4-PZ3)	419	417	417
BRIDGE (T27-PZ9)	546	660	574
AIRPORT (T1-PZ4)	921	868	1090
SEAPORT (T2-PZ0)	870	835	909
FINAL	1116	1389	1390
Latency: All Tasks	27.67	79.24	83.76
AVERAGE	45.44	130.03	143.92
ST. DEV	313	792	793
MAX	0	0	0
MIN			

Figure 4.28: Behavioral Results

All three architectures achieved the goal of completing the mission, however, none of them took the airport prior to the seaport, one of the requirements described in the written scenario. The full resource architecture, A0-6, completed all tasks first, followed by A1-4 and A2-5 with no significant difference. The latency parameters, which is the difference between the time the task first appears and the time the task is started to be processed, were also calculated for each architecture. Latency was added as a parameter of investigation because it was so prevalent in the

reduced resource architectures. Resource contention may cause some tasks to wait, which has a ripple effect of delays through the scenario. It may also cause an internal reordering of task, as tasks that do not have contentious resources precede. As shown in Figure 4.29, an example of the behavioral results of the A1-4 architecture, the contention for the SAT platform, which is necessary to identify enemy, friendly, or neutral threats, was the main cause of delay. The latency builds throughout the course of the scenario in each of the architectures. The nominal A0-6 architecture has very little latency, while the two reduced resource architectures incur much more.

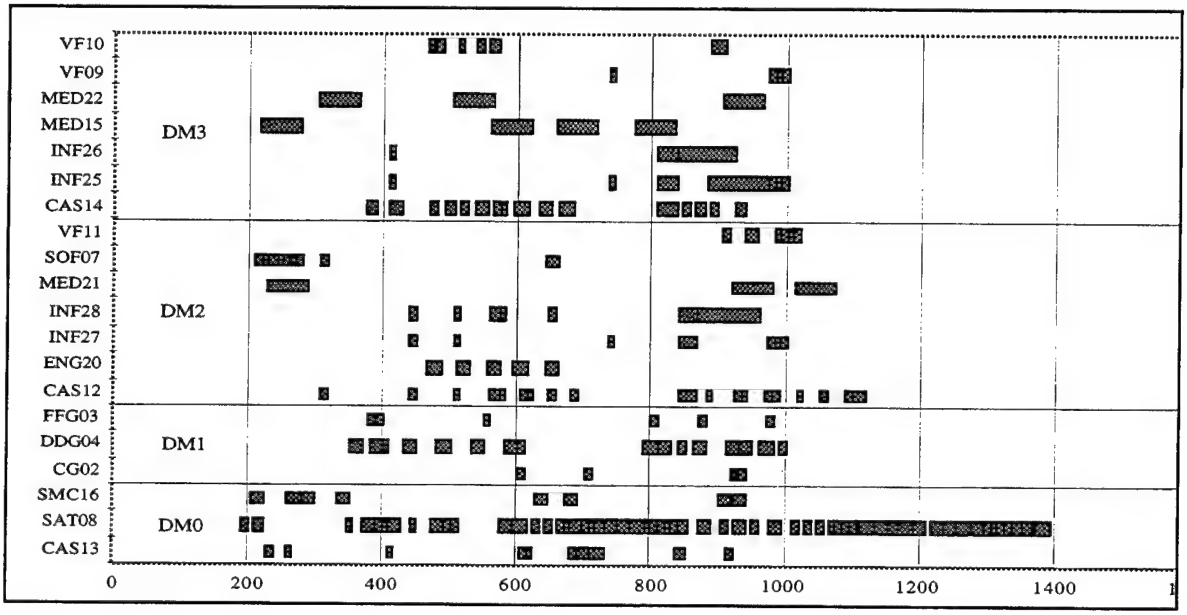


Figure 4.29: A1-4 Platform Usage Results

One of the shortcomings of the model is illustrated in Figure 4.29. The architectures were modeled to complete all tasks regardless of how long they are delayed before they are executed. As shown above, there are tasks being executed after the mission objectives, the securing of the seaport and the airport, are achieved. Post processing of this data would remove these task and deduct the "value" parameter of each task from the architecture's score for this mission. This would be equivalent to having missed this task in the DDD.

Pre-Experimental Modeling Results

The second phase of simulations attempted to stress the architectures outside of the baseline scenario. The first exercise was to change the tempo of inputs in order for an appropriate "time anchor" to be found for the experimental design. The scenario was designed so that it could be

expanded or contracted easily by using a multiplier "h" that controlled the scenario's tempo. If the multiplier was decreased, the tempo of operations increased, and the mission objectives are achieved sooner, if they can be achieved at all. As the tempo of operations decreased, i.e., increasing the multiplier h, there is less and less difference in the performance of the architectures. Even at the baseline factor of h = 10 the differences are very small; at h = 100 there is hardly any difference at all. Note that the values at h = 10 are slightly different than the original behavioral results, due to a second, slightly different task aggregation provided by UConn. These results are shown in Figure 4.30.

	Airport			Seaport		
	A0-6	A1-4	A2-5	A0-6	A1-4	A2-5
h =						
3	378	*	622	430	593	813
5	505	660	729	557	613	680
10	877	840	864	825	802	845
15	1197	1156	1158	1145	1123	1206
30	2157	2125	2130	2105	2109	2103
60	4094	4042	4072	4042	4075	4039
70	4744	4725	4689	4692	4692	4722
90	6044	6025	5989	5992	5992	6022
100	6694	6675	6639	6642	6642	6672

*Model did not complete task at this tempo.

Figure 4.30: Varying Tempo Results

The second exercise was to vary the number of communication channels between DMs. This idea occurred accidentally. In exploring the best way to model the DM messaging, it was noticed that the number of messages allowed per time period had an effect on the completion time under increased tempo of operations depending on the number of decision makers. This raised the question of the relationship between communication resources available and the number of DMs that have to coordinate in each architecture. The communication structure was designed so that all DMs could communicate with all DMs, as indicated in the original experimental design. The results of varying communication channels at a increased tempo (smaller h) are shown in Figure 4.31.

All architectures show some sensitivity in performance to the number of communication channels, however, the nature of the effect is not yet clear. It requires further investigation using the executable model and an accurately representation of the communication structure enforced by the DDD and the experimenters.

	Airport			Seaport		
	A0-6	A1-4	A2-5	A0-6	A1-4	A2-5
h=5						
Com=						
1	505	660	729	557	613	680
2	501	788	707	552	607	658
2	501	851	729	552	660	696
4	501	X	703	552	X	674
5	501	X	X	552	X	X

X= Not an appropriate configuration.

Figure 4.31: Varying Number of Communication Channel Results

4.2.7 Summary of Variables

For this model, the following initial conditions were fixed: the platform initial locations and the platform velocities. To model each specific architecture, the following mapping information was inserted: The platforms owned by each DM, the tasks to be done by each DM, and the platforms required by each task. In order to simulate the executable model and observe the differences between architectures, a fixed scenario of 175 tasks was input to each model. The following outputs of each simulation were recorded: the start and finish time of each task, which DMs and which platforms participated in the task, and, in the case of coordinated tasks, the number of communications between DMs. This recorded data will be used to do post experimental data comparison. Appropriate values can be compared to actual DDD output. By using the DDD task_ids, the mission tasks can be identified and their start and duration times read from the output file.

Based on the modeling results, the following model summarizes the controlled variables that affect the mission completion time.

Delay d = f(Architecture, # of DMs, # of Comm Channels, Tempo)

Where

Architecture: {A0, A1, A2}

# of DMs: {4, 5, 6}

# of Communication Channels: {1,2,3,4,5}

Tempo: {3,5,10,15,30,60,70,90,100}

This results in a sample space of 405 cases (3 x 3 x 5 x 9). However, not all cases are possible. For example the number of communication channels is n-1 where n is the number of decision makers. Also, not all architectures have been considered with 4, 5, or 6 DMs. Consequently, the actual number of cases reduces to 24, all of which have been modeled.

Although GMU and UConn opened a discussion on appropriate performance measures to use during the pre-experimental modeling, they did not yield fruitful results. The performance measures should show differences between the architectures during the simulation. They should also be measures that are observable and measurable in the experiment. This will be a continuing area of research.

4.2.8 Conclusions and GMU's A2C2 Research Progress

GMU has met its objectives of the A2C2 program, third iteration, of participating in a team approach to developing executable, pre-experimental models. As part of the pre-experimental modeling team consisting of GMU, UConn, and NPS, we have a much better understanding of UConn's architectural design process and NPS' experimental implementation. This new knowledge will help us to continue to develop better models that execute the statically designed architectures with the experimental scenario in order to analyze and predict performance and affect hypotheses generation. By using an OO approach and designing re-configurable models we are moving toward modeling true adaptation.

GMU has participated in three iterations of the A2C2 program. For the first experiment GMU produced a limited model after the experiment was conducted in order to provide a proof of concept of an executable model. For the second experiment, GMU produced a set of executable models using the SADT approach. The model design included a full set of static models: process, data, rule, and state transition models as well as a data dictionary. The problem of model concordance was solved with a hyper text approach. They were used to implement an executable CPN model, however, the model was created after the experiment was already completed. GMU has been a main contributor to the third experiment with the development and execution of its pre-experimental models. GMU's progress between the second and third experiment in the A2C2 program are shown in Figure 4.32.

A2C2-2	A2C2-3
Post experiment modeling	Pre experimental modeling
SADT	OO
No Sea Battle	Sea Battle Included
Key Thread Analysis	Performance Measures
GMU Only	Team Approach

Figure 4.32: Comparison of GMU's Models

4.3 Summary

Progress achieved on all tasks during the period has been reported.

5. CHANGES

Changes in the scope of work as a result of the three year renewal of the project have been documented in Sections 2, 3, and 4.

6. CURRENT PERSONNEL

The staff members of the C3 Architectures Laboratory that worked on this project during the reporting period are shown below. The ones in italics are continuing on the project. The others have received their degrees and have moved to jobs with industry.

Prof. Alexander H. Levis	*Principal Investigator*
Dr. Abbas K. Zaidi	*Consultant*
Dr. Didier Perdu	Visiting Res. Scientist
Mr. Lee Wagenhals	*Research Instructor (Ph.D.)*
Ms. Holly H. Handley	*Graduate Res. Asst. (Ph.D.)*
Ms. Zainab Zaidi	*Graduate Res. Assistant (MS)*
Mr. Eric Tsibertzopoulos	Graduate Research Asst. (MS; May 97)
Mr. Alexey Yankovski	Undergraduate Res. Assistant (BS; May 97)
Mr. Tim Taylor	Undergraduate Res. Assistant (to May 97)

8. DOCUMENTATION

1. Hedy L. Rashba (1993). Problems in Concurrency and Coordination in Decision Making Organizations. Report GMU/C3I-143-R, C3I Center, George Mason University, Fairfax, VA.

2. A. H. Levis (1993). Adaptive Decision Making and Coordination in Variable Structure Organizations. Report GMU/C3I-147-IR, C3I Center, George Mason University, Fairfax, VA.

3. Zhenyi Jin (1994). Deadlock and Trap Analysis in Petri Nets. MS Thesis, Systems Engineering Department, George Mason University, Fairfax, VA.

4. A. H. Levis and D. M. Perdu (1994). Object Oriented Design of Decision Making Organizations, *Proc. 1994 The First Workshop on Command Informations Systems,* Oxon, UK. pp. 372-384. Defence Research Agency. Malvern.

5. A. K. Zaidi (1994). Validation and Verification of Decision Making Rules, Report GMU/C3I-155-TH, C3I Center, George Mason University, Fairfax, VA.

6. A. H. Levis and D. M. Perdu (1994). Object Oriented Design of Decision Making Organizations, A. H. Levis & I. S. Levis (Editors), *The Science of Command and Control: Part III, Coping with Change,* AFCEA International Press, Fairfax, VA.

7. A. H. Levis, D. M. Perdu and A. K. Zaidi (1994). Adaptive Decision Making and Coordination in Variable Structure Organizations. Report GMU/C3I-151-IR, C3I Center, George Mason University, Fairfax, VA.

8. A. K. Zaidi and A. H. Levis (1995). Rule Decomposition and Validation for Distributed Decision Making, *Proc. 1995 First International Symposium on Command and Control Research and Technology,* National Defense University, Washington, DC. pp. 210-217.

9. A. K. Zaidi and A. H. Levis (1995). Validation and Verification of Decision Making Rules, *Proc. 1995 6th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Man-Machine Systems,* MIT, Cambridge, MA. pp. 53-60.

10. A. H. Levis, D. M. Perdu & A. K. Zaidi (1995). CAESAR II: Computer Aided Evaluation Of System Architectures, Report GMU/C3I-162-R, C3I Center, George Mason University, Fairfax, VA.

11. A. K. Zaidi and A. H. Levis (1995). Object Oriented Design of a Multilevel Hierarchical Organization Structure, *Proc. US/Portugal Workshop On Underwater Vehicles and Intelligent Control,* Lisbon, Portugal. University of So. Louisianna Press.

12. . K. Zaidi and A. H. Levis (1997).Validation and Verification of Decision Making Rules. *Automatica*, Vol. 33, No. 2, pp. 155 - 169, February 1997.

13. A. K. Zaidi and A. H. Levis (1996). On Generating DIS Architectures Using Genetic Algorithms, Paper GMU/C3I-166-P, C3I Center, George Mason University, Fairfax, VA.

14. Didier M. Perdu and Alexander H. Levis (1996). Object Oriented Design of An Air Combat Operations System Using Eagle Vision, *Proc. of the 1996 C2 Symposium,* Monterey, California, June 24 - 28.

15. Didier M. Perdu and Alexander H. Levis (1996). Distributed Process Coordination in Adaptive Command and Control Teams, *Proc. of the 1996 C2 Symposium,* Monterey, California, June 24 - 28.

16. Dennis M. Buede and Lee W. Wagenhals (1996). Influence Diagram Representation of Dynamic, Distributed Decision Making, *Proc. of the 1996 C2 Research and Technology Symposium,* Monterey, California, June 24 - 28.

17. Lee W. Wagenhals (1996). Digitization of the Battlefield: Approaches for Assessing Behavior and Performance of Distributed Command and Control Systems, *Proc. of the 1996 C2 Research and Technology Symposium,* Monterey, California, June 24 - 28.

18. Didier Leroy and Didier Hoffman (1996). Criteria of C3I Effectiveness, *Proc. of the 1996 C2 Research and Technology Symposium,* Monterey, California, June 24 - 28.

19. Levis, A. H. (1996). Adaptive Decision Making and Coordination in Variable Structure Organizations, Report GMU/C3I-176-IR, C3I Center, George Mason University, , Fairfax, VA.

20. Levis, A. H., D. M. Perdu, E. Tsibertzopoulos and A. Farshadfar (1996). A2C2 - The First Experiment: Architectural Considerations, Report GMU/C3I-177-R, C3I Center, George Mason University, Fairfax, VA.

21. Levis, A. H. & D.M. Perdu (1996). CAESAR II: A System for the Design and Evaluation of Command and Control Organizations, *Proceedings of the 1996 ICCRTS Conference,* Market Bosworth, England, September 23 - 25.

22. Levis, A. H. (1996). System Architectures. *Handbook on Systems Engineering and Management,* A.P. Sage and W.B. Rouse, Eds., Wiley, NY. (to appear in 1997)

23. A. K. Zaidi, "On Temporal Programming using Petri Nets," submitted for publication to *the IEEE Trans. On Systems, Man and Cybernetics*, 6/97.

24. A. K. Zaidi and A. H. Levis, "TEMPER: A Temporal Programmer for Time-Sensitive Control of Discrete Event Systems" submitted for publication to *Discrete Event Dynamic Systems,* 7/97.

25. D. M. Perdu and A. H. Levis, "Adaptation as a Morphing Process: A Methodology for the Design and Evaluation of Adaptive Command and Control Teams," submitted for publication in *Computational and Mathematical Organization Theory,* 8/97

26. A. H. Levis, "Measuring the Effectiveness of C4I Architectures," (1977) Proc. 1997 International Symposium on Defense Information, Korean Institute for Defense Information Systems, Seoul, Republic of Korea.

27. D. M. Perdu and A. H. Levis, (1977) "A Methodology for Adaptive Command and Control Team Design and Evaluation," *Proc. Third Int'l Symposium on Command and Control Research and Technology,* National Defense University, Washington, DC.

28. D. Leroy and A. H. Levis, (1977) "VAUBAN: Validation of Architectures by AMIS and Navigator)," *Proc. Third Int'l Symposium on Command and Control Research and Technology,* National Defense University, Washington, DC.

29. A. H. Levis, "Systems Architectures," in *Wiley Encyclopedia of Electrical and Electronic Engineering,* J. G. Webster, Ed., Wiley (accepted for publication; to appear in 1998)